

Two Puzzles About Computation

Samson Abramsky

1 Introduction

Turing's classical analysis of computation [13] gives a compelling account of the nature of the computational process; of *how* we compute. This allows the notion of *computability*, of what can in principle be computed, to be captured in a mathematically precise fashion.

The purpose of this note is to raise two different questions, which are rarely if ever considered, and to which, it seems, we lack convincing, systematic answers. These questions can be posed as:

- Why do we compute?
- What do we compute?

The point is not so much that we have no answers to these puzzles, as that we have no established body of theory which gives satisfying, systematic answers, as part of a broader understanding. By raising these questions, we hope to stimulate some thinking in this direction.

These puzzles were raised in [2]; see also [3].

2 Why Do We Compute?

The first puzzle is simply stated:

Why do we compute?

By this we mean: why do we perform (or build machines and get them to perform) actual, physically embodied computations?

There is, indeed, an obvious answer to this question:

To gain information (which, therefore, we did not previously have).

But — how is this possible?¹ Two problems seems to arise, one stemming from physics, and one from logic.

Problem 1: Doesn't this contradict the second law of thermodynamics?

Problem 2: Isn't the output *implied* by the input?

We shall discuss each of these in turn.

¹Indeed, I was once challenged on this point by an eminent physicist (now knighted), who demanded to know how I could speak of information increasing in computation when Shannon Information theory tells us that it cannot! My failure to answer this point very convincingly at the time led me to continue to ponder the issue, and eventually gave rise to this discussion.

Problem 1

The problem is that, presumably, information is conserved in the *total* system. The natural response is that, nevertheless, there *can* be information flow between, and information increase in, *subsystems*; just as a body can gain heat from its environment. More precisely, while the entropy of an isolated (total) system cannot decrease, a sub-system *can* decrease its entropy by consuming energy from its environment.

Thus if we wish to speak of information flow and increase, this must be done relative to subsystems. Indeed, the fundamental objects of study should be *open systems*, whose behaviour must be understood in relation to an external environment. Subsystems which can observe incoming information from their environment, and act to send information to their environment, have the capabilities of *agents*.

Moral: Agents and their interactions are intrinsic to the study of information flow and increase in computation. The classical theories of information do not reflect this adequately.

Observer-dependence of information increase? Yorick Wilks (personal communication) has suggested the following additional twist. Consider an equation such as

$$3 \times 5 = 15.$$

The forward direction $3 \times 5 \rightarrow 15$ is obviously a natural direction of computation, where we perform a multiplication. But the reverse direction $15 \rightarrow 3 \times 5$ is also of interest — finding the prime factors of a number! So it seems that the *direction of possible information increase* must be understood as relative to the observer or user of the computation!

Can we in fact find an objective, observer-independent notion of information increase? This seems important to the whole issue of whether information is inherently subjective, or whether it has an objective structure.

Problem 2

The second puzzle is the computational version of what has been called the *scandal of deduction* [8, 6, 11]. The logical problem is to find the sense in which logical deduction can be informative, since, by the nature of the process, the conclusions are ‘logically contained’ in the premises. So what has been added by the derivation? This is a rather basic question, which it is surprisingly difficult to find a satisfactory answer to.

Computation can be modelled faithfully as deduction, whether in the sense of deducing the steps that a Turing machine takes starting from its initial configuration, or more directly via the Curry-Howard isomorphism [5, 9], under which computation can be viewed as performing cut-elimination on proofs, or normalization of proof terms. Thus the same question can be asked of computation: since the result of the computation is logically implied by the program together with the input data, what has been added by computing it?

The same issue can be formulated in terms of the logic programming paradigm, or of querying a relational database: in both cases, the result of the query is a logical consequence of the data- or knowledge-base.

It is, of course, tempting to answer in terms of the complexity of the inference process; but this seems to beg the question. We need to understand first what the inference process is doing for us!

We can also link this puzzle to another well-known issue in logic, namely the principle of *logical omniscience* in epistemic logic, which is unrealistic yet hard to avoid. This principle can be formulated as follows:

$$[K_a\phi \wedge (\phi \rightarrow \psi)] \rightarrow K_a\psi.$$

It says that the knowledge of agent a is deductively closed: if a knows a proposition ϕ , then he knows all its logical consequences. This is patently untrue in practice, and brings us directly back to our puzzle concerning computation. We compute to gain information we did not have. We start from the information of knowing the program and its input, and the computation provides us with explicit knowledge of the output. But what does ‘explicit’ mean?

The computational perspective may indeed provide a usefully clarifying perspective on the issue of logical omniscience, since it provides a context in which the distinction between ‘explicit’ and ‘implicit’ knowledge can be made precise. Let us start with the notion of a function. In the 19th century, the idea of a function as a ‘rule’ — as given by some defining expression — was replaced by its ‘set-theoretic semantics’ as a set of ordered pairs. In other terminology, a particular defining expression is an *intensional description* of a function, while the set of ordered pairs which it denotes is its *extension*.

A program is exactly an intensional description of a function, with the additional property that this description can be used to explicitly calculate outputs from given inputs in a stepwise, mechanical fashion.² We can say that implicit knowledge, in the context of computation, is knowledge of an intensional description; while explicit knowledge, of a data item such as a number, amounts to possessing the *numeral* (in some numbering system) corresponding to that number; or more generally, to possessing a particular form of intensional description which is essentially isomorphic to the extension.

The purpose of computation in these terms is precisely to convert intensional descriptions into extensional ones, or implicit knowledge of an input-output pair into explicit knowledge. The *cost* of this process is calibrated in terms of the resources needed — the number of computation steps, the workspace which may be needed to perform these steps, etc. Thus we return to the usual, ‘common-sense’ view of computation. The point is that it rests on this distinction between intension and extension, or implicit vs. explicit knowledge.

Another important aspect of why we compute is *data reduction*—getting rid of a lot of the information in the input. Note that normal forms are in general *unmanagably big* [14]. Note also that it is *deletion of data* which creates thermodynamic cost in computation [10]. Thus we can say that much (or all?) of the actual usefulness of computation lies in getting rid of the hay-stack, leaving only the needle.

The challenge here is to build a useful theory which provides convincing and helpful answers to these questions. In our view these puzzles, naive as they are, point to some natural questions which a truly comprehensive theory of computation, incorporating a ‘dynamics of information’, should be able to answer.

3 What Do We Compute?

The classical notion of computability as pioneered by Turing [13] focusses on the key issue of *how* we compute; of what constitutes a computation. However, it relies on pre-existing notions from mathematics as to *what* is computed: numbers, functions, sets, etc.

²We refer e.g. to [7, 12] for attempts to give a precise mathematical characterization of ‘mechanical’.

This idea also served computer science well for many years: it is perfectly natural in many situations to view a computational process in terms of computing an output from an input. This computation may be deterministic, non-deterministic, random, or even quantum, but essentially the same general paradigm applies.

However, as computation has evolved to embrace diverse forms and purposes: distributed, global, mobile, interactive, multi-media, embedded, autonomous, virtual, pervasive, ... the adequacy of this view has become increasingly doubtful.

Traditionally, the *dynamics* of computing systems — their unfolding behaviour in space and time — has been a mere means to the end of computing the function which specifies the algorithmic problem which the system is solving.³ In much of contemporary computing, the situation is reversed: the *purpose* of the computing system is to exhibit certain behaviour. The *implementation* of this required behaviour will seek to reduce various aspects of the specification to the solution of standard algorithmic problems.

What does the Internet compute?

Surely not a mathematical function ...

Why Does It Matter?

We shall mention two basic issues in the theory of computation which become moot in the light of this issue.

There has been an enormous amount of work on the first, namely the theory of concurrent processes. Despite this huge literature, produced over the past four decades and more, no consensus has been achieved as to what processes *are*, in terms of their essential mathematical structure. Instead, there has been a huge proliferation of different models, calculi, semantics, notions of equivalence. To make the point, we may contrast the situation with the λ -calculus, the beautiful, fundamental calculus of functions introduced by Church at the very point of emergence of the notion of computability [4]. Although there are many variants, there is essentially a unique, core calculus which can be presented in a few lines, and which delineates the essential ideas of functional computation. In extreme contrast, there are a huge number of process calculi, and none can be considered as definitive.

Is the notion of process too amorphous, too open to different interpretations and contexts of use, to admit a unified, fundamental theory? Or has the field not yet found its Turing? See [1] for an extended discussion.

The second issue follows on from the first, although it has been much less studied to date. This concerns the Church-Turing thesis of universality of the model of computation. What does this mean when we move to a broader conception of what is computed? And are there any compelling candidates? Is there a widely accepted universal model of interactive or concurrent computation?

As a corollary to the current state of our understanding of processes as described in the previous paragraphs, there is no such clear-cut notion of universality. It is important to understand what is at issue here. If we are interested in the process of computation itself,

³Insofar as the dynamics has been of interest, it has been in quantitative terms, counting the resources which the algorithmic process consumes — leading of course to the notions of algorithmic complexity. Is it too fanciful to speculate that the lack of an adequate structural theory of processes has been an impediment to fundamental progress in complexity theory?

the structure of interactive behaviour, then on what basis can we judge if one such process is faithfully simulated by another? It is not of course that there are no candidate notions of this kind which have been proposed in the literature; the problem, rather, is that there are far too many of them, reflecting different intuitions, and different operational and application scenarios.

Once again, we must ask: is this embarrassing multitude of diverse and competing notions a necessary reflection of the nature of this notion, or may we hope for an incisive contribution from some future Turing which will unify and organize the field?

References

- [1] S. Abramsky. What are the Fundamental Structures of Concurrency? We still don't know! *Electronic Notes in Theoretical Computer Science*, 162:37–41, 2006.
- [2] S. Abramsky. Information, processes and games. In J. van Benthem and P. Adriaans, editors, *Handbook of the Philosophy of Information*, pages 483–549. Elsevier Science Publishers, Amsterdam, 2008.
- [3] P. Adriaans and P. van Emde Boas. Computation, information, and the arrow of time. In S.B. Cooper and A. Sorbi, editors, *Computability in Context*, pages 1–18. World Scientific, 2011.
- [4] A. Church. *The Calculi of Lambda Conversion*, volume 6 of *Annals of Mathematics Studies*. Princeton University Press, 1941.
- [5] H.B. Curry, R. Feys, and W. Craig. *Combinatory Logic: Volume 1*. North-Holland, 1958.
- [6] M. D'Agostino and L. Floridi. The enduring scandal of deduction. *Synthese*, 167(2):271–315, 2009.
- [7] R. Gandy. Church's thesis and principles for mechanisms. *Studies in Logic and the Foundations of Mathematics*, 101:123–148, 1980.
- [8] J. Hintikka. Information, deduction, and the a priori. *Nous*, 4(2):135–152, 1970.
- [9] W.A. Howard. The formulae-as-types notion of construction. *To HB Curry: essays on combinatory logic, lambda calculus and formalism*, pages 479–490, 1980.
- [10] R. Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 5:183–191, 1961.
- [11] S. Sequoia-Grayson. The scandal of deduction. *Journal of Philosophical Logic*, 37(1):67–94, 2008.
- [12] W. Sieg. Calculations by man and machine: Mathematical presentation. In *In the scope of logic, methodology, and philosophy of science: volume two of the 11th International Congress of Logic, Methodology and Philosophy of Science, Cracow, August 1999*, page 247. Kluwer Academic Publishers, 2002.
- [13] AM Turing. On computable numbers. *Proceedings of the London Mathematical Society*, 2(42):230–65, 1937.

- [14] Sergei G. Vorobyov. The “hardest” natural decidable theory. In *Proceedings of the 12th Symposium on Logic in Computer Science*, pages 294–305. IEEE Press, 1997.